2048 Klon in C++

Lena Schliebitz

July 21, 2023

Einführung

Der vorliegende Quellcode ist die Implementierung eines einfachen Terminal-Spiels namens "2048". Das ursprüngliche 2048-Spiel ist ein beliebtes Puzzlespiel, bei dem der Spieler Kacheln mit Potenzen von 2 kombiniert, um eine Kachel mit dem Wert 2048 zu erreichen.

Ncurses

Ncurses ist eine simple Bibliothek zur Entwicklung von text-basierten Nutzeroberflächen. Es bietet Funktionen, die im Gegensatz zum standard Cout nicht nur zeilenweise Text herunter schreiben, sondern auch die Möglichkeit bieten, den Text an beliebigen Stellen im Terminal zu platzieren.

Zusätzlich bietet Ncurses mit `WINDOW`s eine Abstraktion des Terminals, in die performant gezeichnet werden kann. Der Inhalt einer WINDOW Instanz wird erst dann auf den Bildschirm geschrieben, wenn die Funktion `wrefresh()` aufgerufen wird. Dadurch wird das Flackern des Bildschirms verhindert, da der Inhalt des Terminals nicht bei jedem Schreibvorgang neu gezeichnet wird.

Datenstrukturen

Das Spielfeld wird für die Spiele-Algorithmen als 4x4 Integer Matrix repräsentiert. Für die Darstellung des Spielfeldes wird eine 4x4 Matrix aus `WINDOW` Instanzen verwendet. Die Render-Funktionen bilden die Integer Matrix auf die `WINDOW` Matrix ab.

Spiel Logik

Die Eingabe des Spiels besteht in der Auswahl einer von 4 Richtungen. Da immer nur eine Richtung aktiv ist, können die Zeilen (oder Spalten) des Bretts getrennt betrachtet werden. Durch Transponieren und Umkehren der Matrix

können wir jede Richtung auf die Verarbeitung Spalten von oben nach unten zurückführen.

Die Funktion `processMove` kombiniert die Kacheln in einer einzelnen Spalte von oben nach unten. Zuerst werden mithilfe der Funktion `stackattheend` alle nicht leeren Kacheln am unteren Ende der Spalte gestapelt. Dadurch wird das anschließende Kombinieren der Kacheln vereinfacht, da nur direkt benachbarte Felder betrachtet werden müssen. Befinden sich zwei gleiche Zahlen nebeneinander, so wird die Zahl in der einen Kachel verdoppelt, in der anderen auf Null gesetzt. Somit erhalten wir eine leere Kachel. Nach dem Kombinieren werden die Kacheln erneut gestapelt, um eventuell entstandene Lücken zu schließen.

Nach jedem Zug wird eine neue Kachel mit dem Wert 2 an einer zufälligen Position auf dem Spielfeld platziert. Die Funktion `spawnNewNumber` wählt so lange eine zufällige Position auf dem Spielfeld, bis ein leeres Feld (repräsentiert durch 0) gefunden wird und platziert dort eine 2.

Die Funktion `hasLost` prüft, ob noch gültige Züge möglich sind. Dazu wird geprüft, ob es noch leere Felder gibt, oder ob zwei benachbarte Kacheln den gleichen Wert haben. Dies wird für Spalten und Zeilen durchgeführt. Wenn dies nicht der Fall ist, hat der Spieler verloren.

Code Struktur

Jede Funktion sollte nur eine Aufgabe erfüllen. Dadurch wird der Code übersichtlicher und leichter zu warten. Eine Funktion soll entweder eine Ausgabe erzeugen oder nur für ihre Seiteneffekte ausgeführt werden. Da nur 2 nennenswerte Datenstrukturen verwendet werden, ist es nicht nötig, diese objektorientiert zu behandeln.

Input loop

Das Spiel läuft in einer Endlosschleife, die bei jedem Durchlauf die Benutzereingabe abfragt und das Spielfeld entsprechend der Eingabe aktualisiert. Zur Verarbeitung von Richtungen wird ein enum verwendet, das die gültigen Richtungen enthält und Lesbarkeit sowie Tool-Unterstützung verbessert. Die Endlosschleife wird durch die Funktion `hasLost` unterbrochen, die prüft, ob der Spieler noch Züge machen kann. Wenn dies nicht der Fall ist, wird die Schleife unterbrochen und das Spiel ist vorbei. Der Haupt-Input loop selbst läuft ebenfalls in einer Endlosschleife, die nur durch das Drücken der Taste `q` unterbrochen wird. Dadurch kann sich das Spiel nachdem der Spieler verloren hat selbst neu starten.